Please amend claims 29, 31 and 32, and insert claims 33-38 as follows:

1. A method for programmable processing in a computer graphics pipeline, comprising:

(a) receiving data from a source buffer;

(b) performing programmable operations on the data in order to generate output, wherein the operations are programmable by a user utilizing instructions from a predetermined instruction set;

(c) storing the output in a register; and

(d) wherein the output stored in the register is used in performing the programmable operations on the data.

2. The method as recited in claim 1, wherein only one vertex is processed at a time.

3. The method as recited in claim 1, wherein operations (a)-(d) are processed for multiple vertexes in parallel.

4. The method as recited in claim 1, wherein the data includes a constant.

5. The method as recited in claim 4, wherein the constant is stored in a constant source buffer.

6. The method as recited in claim 5, wherein the constant is accessed in the constant source buffer using an absolute or relative address.

7. The method as recited in claim 1, wherein the data includes vertex data.

8. The method as recited in claim 1, wherein the register has single write and triple read access.

9. The method as recited in claim 1, and further comprising storing the output in a destination buffer.

10. The method as recited in claim 9, wherein the output is stored in the destination buffer under a predetermined reserved address.

11. The method as recited in claim 1, and further comprising negating the data.

12. The method as recited in claim 1, and further comprising swizzling the data.

13. A computer program embodied on a computer readable medium for programmable processing in a computer graphics pipeline, comprising:

(a) a code segment for receiving data from a source buffer;

(b) a code segment for performing programmable operations on the data in order to generate output, wherein the operations are programmable by a user utilizing instructions from a predetermined instruction set;

(c) a code segment for storing the output in a register; and

(d) wherein the output stored in the register is used in performing the programmable operations on the data.

14. The computer program as recited in claim 13, wherein only one vertex is processed at a time.

15. The computer program as recited in claim 13, wherein the code segments are executed for multiple vertexes in parallel.

16. The computer program as recited in claim 13, wherein the data includes a constant.

17. The computer program as recited in claim 16, wherein the constant is stored in a constant source buffer.

18. The computer program as recited in claim 17, wherein the constant is accessed in the constant source buffer using an absolute or relative address.

19. The computer program as recited in claim 13, wherein the data includes vertex data.

20. The computer program as recited in claim 13, wherein the register has single write and triple read access.

21. The computer program as recited in claim 13, and further comprising a code segment for storing the output in a destination buffer.

22. The computer program as recited in claim 21, wherein the output is stored in the destination buffer under a predetermined reserved address.

23. The computer program as recited in claim 13, and further comprising a code segment for negating the data.

24. The computer program as recited in claim 13, and further comprising a code segment for swizzling the data.

25. A system for programmable vertex processing, comprising:

(a) a source buffer for storing data;

(b) a functional module coupled to the source buffer for performing programmable operations on the data received therefrom in order to generate output, wherein the operations are programmable by a user utilizing instructions from a predetermined instruction set; and

(c) a register coupled to the functional module for storing the output such that the output may be used by the functional module in performing the programmable operations on the data.

26. A method for performing an operation on data in a computer graphics pipeline, comprising:

(a) receiving a source location identifier indicating a source location of data to be processed, wherein the source location includes a plurality of components;

(b) receiving a source component identifier indicating in which of the plurality of components of the source location the data resides;

(c) retrieving the data based on the source location identifier and the source component identifier;

(d) performing an operation on the retrieved data in order to generate output;

(e) identifying a destination location identifier indicating a destination location of the output, wherein the destination location includes a plurality of components;

(f) identifying a destination component identifier indicating in which of the plurality of components of the destination location the output is to be stored; and

(g) storing the output based on the destination location identifier and the destination component identifier.

27. The computer program as recited in claim 26, wherein the operation is selected from the group consisting of a no operation, address register load, move, multiply, addition, multiply and addition, reciprocal, reciprocal square root, three component dot product, four component dot product, distance vector, minimum, maximum, set on less than, set on greater or equal than, exponential base two (2), logarithm base two (2), and/or light coefficients.

28. A computer-readable medium containing a data structure for performing an operation on data in a computer graphics pipeline, comprising:

(a) a source location identifier indicating a source location of data to be processed, wherein the source location includes a plurality of components;

(b) a source component identifier indicating in which of the plurality of components of the source location the data resides, wherein the data is retrieved based on the source location identifier and the source component identifier for performing an operation on the retrieved data in order to generate output;

(c) a destination location identifier indicating a destination location of the output, wherein the destination location includes a plurality of components; and

(d) a destination component identifier indicating in which of the plurality of components of the destination location the output is to be stored, wherein the output is stored based on the destination location identifier and the destination component identifier.

29. (Amended) A method for programmable processing in a computer graphics pipeline, comprising:

(a) receiving graphics data;

(b) determining whether the graphics pipeline is operating in a programmable mode;

(c) performing programmable operations on the graphics data in order to generate output if it is determined that the graphics pipeline is operating in the programmable mode, wherein the operations are programmable by a user utilizing instructions from a predetermined instruction set which is different from an instruction set of a standard graphics application program interface in order to provide increased flexibility in programming the processing in the computer graphics pipeline; and

(d) performing operations on the graphics data in order to generate output in accordance with [a] the standard graphics application program interface if it is determined that the graphics pipeline is not operating in the programmable mode in order to preserve hardware optimization afforded by the standard graphics application program interface.

30. The method as recited in claim 29, wherein the standard graphics application program interface includes OpenGL®.

31. (Amended) The method as recited in claim 29, wherein the graphics data includes vertex data.

32. (Amended) A computer program embodied on a computer readable medium for programmable processing in a computer graphics pipeline, comprising:

(a) a code segment for receiving graphics data;

(b) a code segment for determining whether the graphics pipeline is operating in a programmable mode;

(c) a code segment for performing programmable operations on the graphics data in order to generate output if it is determined that the graphics pipeline is operating in the programmable mode, wherein the operations are programmable by a user utilizing instructions from a predetermined instruction set which is different from an instruction set of a standard graphics application program interface in order to provide increased flexibility in programming the processing in the computer graphics pipeline; and

(d) a code segment for performing operations on the graphics data in order to generate output in accordance with [a] the standard graphics application program interface if it is determined that the graphics pipeline is not operating in the programmable mode in order to preserve hardware optimization afforded by the standard graphics application program interface.

33. (New) The method as recited in claim 29, wherein the programmable operations are performed using a system comprising:

(a) a source buffer for storing the graphics data;

(b) a functional module coupled to the source buffer for performing the programmable operations on the graphics data received therefrom in order to generate the output; and

(c) a register coupled to the functional module for storing the output such that the output may be used by the functional module in performing the programmable operations on the graphics data.

34. (New) The method as recited in claim 29, wherein the programmable operations are performed using a data structure comprising:

(a) a source location identifier indicating a source location of the graphics data to be processed, wherein the source location includes a plurality of components;

(b) a source component identifier indicating in which of the plurality of components of the source location the graphics data resides, wherein the graphics data is retrieved based on the source location identifier and the source component identifier for performing an operation on the retrieved graphics data in order to generate the output;

(c) a destination location identifier indicating a destination location of the output, wherein the destination location includes a plurality of components; and

(d) a destination component identifier indicating in which of the plurality of components of the destination location the output is to be stored, wherein the output is stored based on the destination location identifier and the destination component identifier.

35. (New) The method as recited in claim 29, wherein the instructions from the predetermined instruction set are selected from the group consisting of a no operation, address register load, move, multiply, addition, multiply and addition, reciprocal, reciprocal square root, three component dot product, four component dot product, distance vector, minimum, maximum, set on less than, set on greater or equal than, exponential base two (2), logarithm base two (2), and/or light coefficients.

36. (New) The computer program as recited in claim 32, wherein the programmable operations are performed using a system comprising:

(a) a source buffer for storing the graphics data;

(b) a functional module coupled to the source buffer for performing the programmable operations on the graphics data received therefrom in order to generate the output; and

(c) a register coupled to the functional module for storing the output such that the output may be used by the functional module in performing the programmable operations on the graphics data.